AMENDMENT TO THE SPECIFICATION

Please amend the paragraph beginning on page 5, line 20 as follows:

An aspect of the present invention is to allow the system to automatically adapt the dialogue flow so that it stays focused on the user's most recent inputs[[[RL1]]]. Generally, whenever recognition results are received, this information is retained in a manner so as to provide an order indicating the relative order it was received. In this manner, the most recently recognition results can be identified. In one embodiment, memory is used in the form of a "stack". The stack comprises identifiers related to recognition results received. When the dialog is created, it looks for controls related to the recognition results identified at the top of the stack, for example, whether this recognition result needs to be confirmed. Although the controls typically include means such as an attribute to indicate a selected order for execution, this means that controls later in the selected order than others can be "promoted" and run before them, provided that they are related to the top-most of the stack whereas the others are not.

Please amend the paragraph beginning on page 7, line 14 as follows:

FIG. 14 is a method for comparing a SemanticItem on the focus stack with answers or confirms related to a QA.[[[RL2]]]

Please amend the paragraph beginning on page 75, line 6 as follows:

A system wide [[[RL3]]] "focussing" value can be used to identify if focussing is to be performed or not. With focussing, the general algorithm described above can be represented as follows, where other portions are the same.

Please amend the paragraph beginning on page 75, line 12 as follows:

1. If focussing is desired and the stack is not null (indicating focused SemanticItems [[[RL4]]] are present), find the first active (as defined above) QA [[[RL5]]] corresponding to the SemaniticItem at the top of the stack.

Please amend the paragraph beginning on page 76, line 8 as follows:

It should also be noted in a further embodiment, that the mechanism or manner in which the most recent SemanticItems have been saved need not ensure that they all be saved indefinitely. Rather, referring to FIG. 13[[_]]-where the stack 450 , illustrated by way of example for storing such information, is some selected or finite length such that when the stack 450 is full and a further SemanticItem placed thereon, it causes the lowest or oldest SemanticItem to be pushed [[[RL6]]] off. This technique may be convenient in that the dialogue created thereby will not return to some item of information that the user spoke long ago.

Please amend the paragraph beginning on page 77, line 13 as follows:

At step 460 the most recent SemanticItem is identified from the stack 450. At step 462 , each QA control's corresponding answers are compared for a relation to the most recent SemanticItem. If a related answer is found, the QA control considered for execution. This is subject to the QA being active under the usual activation conditions (semantic item should be empty, etc.) [[[RL7]]]

Please amend the paragraph beginning on page 77, line 21 as follows:

If at step 462 no QA control is found based on related answers, the method continues at step 464 whereat each QA control's corresponding confirms are compared for a relation to the most recent SemanticItem. If a related confirm is found the QA control is considered for execution. [[[RL8]]]

Please amend the paragraph beginning on page 79, line 1 as follows:

In the second case, the system should confirm the departure city first and then ask about the destination city second. If the user provided both the departure city and the destination city, the application author may want to confirm the departure city first and then the arrival city. Writing such a dialogue flow without using automated focusing is time consuming and error prone. Using the focusing mechanism, the application author does not have to worry about getting the ordering right [[[RL9]]]. (However, the application author will still need to worry about some of the ordering (e.g., which city to confirm when both are given at the same time) although

not as much as before.) If the user provides the departure city, the associated SemanticItem is pushed on the stack. When the control algorithm RunSpeech looks for a suitable QA control to execute, only QA controls related to that SemanticItem will be considered. In this case the QA confirming the departure city will be executed, even if the QA asking for the destination city comes earlier in speech index order.

Please amend the paragraph beginning on page 81, line 5 as follows:

FIG. [[13]]15 pictorially illustrates information to be gathered organized as "topics" in order to execute portions of the dialog. In the illustrative example, the topics pertain to a travel site that allows users to input information related to a departure city 500 , a departure date and time 502 , an arrival city 504 and an arrival date and time 506. Each topic 500 , 502 , 504 and 506 comprises a collection of one or more questions, answers, commands or validators such as illustrated in FIG. 11 to form a dialog for each corresponding topic. Each of the collections 500 , 502 , 504 and 506 includes a label or identifier 500 A, 502 A, 504 A and 506 A, respectively. In addition, the collections can be grouped in two or more sets also identified by a label or identifier. In the illustrated example, a larger collection is identified as 508 A (representative of the complete page), which comprises the collections 500 , 502 , 504 and 506 as a group or hierarchy.

Please amend the paragraph beginning on page 82, line 23 as follows:

In the voice-only mode scenario, the "Collection Identifier" can be set to identifier 508 A, in which case the control algorithm will execute in the manner discussed above for the voice-only mode of operation to execute the complete dialog. However, it should be noted that in an alternative embodiment, a separate manager algorithm or module 512 can be used to individually and sequentially activate each of the collections 500 , 502 , 504 and 506 . In the example illustrated in FIG. [[13]]15, the manager algorithm 512 would issue a command identifying that collection 502 is active, whereupon after control algorithm executes the dialog of collection 502 , control returns back to the manager algorithm 512 . The manager algorithm 512 would then issue a command identifying that collection 504 is now active. This process is repeated until each of

the collections 500 , 502 , 504 and 506 have been activated as prescribed by the manager algorithm 512.

Please amend the paragraph beginning on page 83, line 22 as follows:

FIG. [[14]]16 illustrates an exemplary display rendering 520 for the travel page. As data is entered, more collections may become active or may not be relevant and thus deactivated. As described above, each collection 500 , 502 , 504 and 506 has an associated identifier or label 500 A, 502 A, 504 A and 506 A in the dialog. Each textbox in the page provided to the client also has an associated identifier or label (not shown). When a textbox, or other button such as button 522 associated with each textbox, is activated by the user, a simple function can then be called using the label or identifier of the textbox or other button 522 as an input parameter to identify the corresponding collection identifier that the selected textbox or button 522 is associated with. With the collection identifier determined, the control algorithm can be executed upon the dialog of the corresponding collection, for example, by issuing the J-script command provided above.

Please amend the paragraph beginning on page 85, line 25 as follows:

However, in the multi-modal scenario, the input speech that has been recognized is generally rendered in the textbox that has been selected. Again, based upon automatic confirmation or whether or not a confidence threshold has been exceeded, a confirmation prompt can be displayed asking whether or not the input speech has been properly recognized. In FIG. [[14]]16, a confirmation prompt is illustrated at 530 in response to a confidence threshold not being exceeded. In this example, the prompt 530 includes a "yes" button 532 and a "no" button 534 . The confirmation prompt could be rendered audibly and/or the user's response could be provided by buttons 532 and 534 or via speech input.

Please amend the paragraph beginning on page 86, line 27 as follows:

Also, audible rendering of prompts typically includes active monitoring of the length of silence that may result after rendering the audible prompt. If a sufficient amount of silence occurs after playing the prompt, the user may not have heard the prompt and some methods are then employed to continue the dialog, for example, by replaying the prompt. However, when a prompt

is displayed, such as the confirmation 530 of FIG. [[14]]16, the notion of time may be irrelevant. Thus, if a prompt is being displayed rather than being rendered audibly, it may be appropriate to disable or otherwise modify silence measuring methods occurring after rendering. In effect, the control algorithm could suspend the dialog after displaying a prompt, such as confirmation prompt 530 , and begin again with the user selecting one of the buttons "yes" or "no" in the prompt 530 . Depending on if any measurement of inactivity is being monitored by the control algorithm, activation of the buttons "yes" or "no" could employ a method to resume processing of the dialog by the control algorithm.

Please amend the paragraph beginning on page 87, line 19 as follows:

Referring back to FIG. [[14]]16, if the user indicates that the recognition was correct, the recognized input can be rendered in the textbox, whereas if the user indicates that the recognition was incorrect, the recognized input would be discarded. However, in the multi-modal scenario, it is also desirable to allow the user to ignore answering the displayed or otherwise rendered confirmation question, and proceed to another textbox or operation with respect to the form provided on the client device. Thus, it is desirable to allow the control algorithm to stop execution of the portion of the control algorithm with respect to the textbox selected if the user does not respond to the confirmation response. Depending on the author of the application, the author may choose to ignore the input speech provided to the textbox, or accept it with the intent of reconfirming its accuracy at a later time.

Please amend the paragraph beginning on page 89, line 27 as follows:

Exposing the SemanticItems and the status information, also allows the application author to reset selected portions of the dialog information as desired during execution. For instance, a reset button can be provided for the each portion of the dialog separately based on the collections 500 , 502 , 504 and 506 . These buttons are indicated at 550 , 552 , 554 and 556 , respectively. If the user, activates any of the buttons 550 , 552 , 554 or 556 , the control algorithm can ascertain which collection has been identified for resetting using a simple function (in a manner similar to that discussed above with respect to identifying textboxes) and then remove the corresponding values in the associated SemanticItems for that collection and/or change the

status information to signify that new data is required. Although in FIG. [[14]]16 use of the reset buttons 550 , 552 , 554 and 556 may appear simplistic, it should be noted the embodiment illustrated is merely exemplary, and that the concept can be applied to more complex data entry forms where resetting would involve more textboxes or complexity.

Please amend the paragraph beginning on page 90, line 19 as follows:

In the multi-modal scenario where the user has the ability to select a textbox for entry of data, typically the user will know initially what the form is asking for due to the fact that there is a visual prompt associated with the textbox. For example, it is clear in FIG. [[14]]16 that textbox 536 is for the "Departure City". Thus, although the control algorithm is running the dialogue associated with the textbox in the multi-modal scenario, there is probably no need to ask the initial prompt in the collection 500 , as would be necessary in the voice-only scenario, since the user probably is provided with a visual prompt. However, it is also desirable that the page be renderable in either a multi-modal scenario or a voice-only scenario. Therefore, the control algorithm can maintain in memory status information as to whether or not the user is operating in a multi-modal scenario, or whether the page is being rendered in a voice-only scenario such as through voice browser 216 . In one embodiment, information can be easily ascertained by the client device upon receipt of the form to determine the appropriate mode of entry. For example, if the client device is a voice browser 216, the mode of entry needs to be operation of the control algorithm in a voice-only mode. The voice-only mode is particularly suitable for focusing as described above. However, if the client device is a multi-modal device such as a PDA, then the mode of entry can be optionally defaulted to a preselected mode of entry as well as be changed such as through activation of a button 540.